Paper

# Using Polymatrix Extensive Stackelberg Games in Security – Aware Resource Allocation and Task Scheduling in Computational Clouds

Agnieszka Jakóbik[1] and Andrzej Wilczyński[1,2]

[1] *Tadeusz Kościuszko Cracow University of Technology, Cracow, Poland*
[2] *AGH University of Science and Technology, Cracow, Poland*

**Abstract—In this paper, the Stackelberg game models are used for supporting the decisions on task scheduling and resource utilization in computational clouds. Stackelberg games are asymmetric games, where a specific group of players' acts first as leaders, and the rest of the players follow the leaders' decisions and make their decisions based on the leader's actions. In the proposed model, the optimal schedules are generated under the security criteria along with the generation of the optimal virtual machines set for the scheduled batch of tasks. The security criteria are defined as security requirements for mapping tasks onto virtual machines with specified trust level. The effectiveness of the proposed method has been verified in the realistic use cases with in the cloud environment with OpenStack and Amazon Cloud standards.**

***Keywords—cloud computing, resource optimization, Stackelberg equilibrium, Stackelberg games.***

## 1. Introduction

The problem of optimal resource allocation and utilization in computational clouds (CC) remains challenging research task in today's parallel computing. The considered computational infrastructure contains a large set of virtual machines (VMs) implemented in physical resources in distributed private and public cloud clusters. In public clouds, the customer can use any number of VMs and pay for used resources. The cloud provider collects the tasks from the users, analyzes the customers' requirements and tries to allocate the virtual and physical resources based on such requirements. Therefore, the users' requests for the access to the cloud services and resources may be dynamic. The users may change the number of VMs, which can be utilized. Therefore, the system must ensure the proper scheduling policies for such dynamic environment.

Task scheduling in the dynamic cloud environments is a complex process of contains multiple stages. Tasks may be scheduled as batch with (dependent) or without (independent) correlations among them [1]. In the scheduling process, the following special security-related users' requirements may be taken into account: selection of an appropriate amount of resources for utilization, safety, security, and intrusion detection. It may be also assumed that idle time for the resource should be minimal. Such conditions improve the energy awareness in scheduling, tasks execution, and building of the green cloud architectures. The set of VMs may be selected based on the estimation of the cost of scheduling, maximal utilization of the memory, and bandwidth, service access, etc.

This paper proposes a new model of selection and allocation of VMs in batch scheduling. In that model we focus on cloud services select based on the users' requirements. The scheduling model is based on central scheduling unit. The polymatrix extensive Stackelberg games are used for calculation of the optimal choosing strategies for the available cloud resources.

The model has been verified in a simple experimental analysis based on use cases specified for two sets of VMs implemented in the cloud clusters in OpenStack Racspace platform and Amazon Cloud.

The paper is organized as follows. In Section 2 the backgrounds of virtualization of the cloud resources and task scheduling in cloud systems are presented. In Section 3 security characteristics are defined, which must provide each of cloud environment. In Sections 4 and 5 related work is reviewed, considered problems are defined and tasks representation and models are explained. In Section 6 we present polymatrix extensive Stackelberg Games models. In Sections 7 and 8 the application of the Stackelberg Games as the support of the decision processes in the optimization of the VMs parameters are presented. In Section 9 the results of the application of the proposed model in realistic use case are shown. Finally, the paper is concluded in Section 10.

## 2. VMs Optimization and Task Scheduling in CC Systems

In CC infrastructure, VMs are the main target "resources" available and selected for scheduling of the computational tasks. They may be created from the images of VMs on demand. VMs can be configured with the characteristics specified in Tables 1 and 2. Tasks are mapped into VMs by using various scheduling procedures. In the case of batch

scheduling, the tasks are grouped into batches. For each consecutive batch, a different workload can be calculated based on the number of atomic numerical operations per second estimated for each task in the batch. It can be interpreted as task complexity measure.

Table 1
Selected Amazon Cloud VM instances

| Model | vCPUs | CPU credits per hour | Mem [GB] |
|---|---|---|---|
| t2.nano | 1 | 3 | 0.5 |
| t2.small | 1 | 12 | 2 |
| t2.xlarge | 4 | 54 | 16 |
| t2.2xlarge | 8 | 81 | 32 |
| m4.16xlarge | 64 | 120 | 256 |

Table 2
Selected OpenStack cloud VM instances

| Flavor | vCPUs | Disk [GB] | RAM [GB] |
|---|---|---|---|
| General1-1 | 1 | 20 SSD | 1 |
| General1-2 | 2 | 40 SSD | 2 |
| General1-4 | 4 | 80 SSD | 4 |
| General1-8 | 8 | 160 | 8 |

In CC, the pay-as-you-go model of provisioning of the cloud resources is used. It means that the cloud user pays only for the utilization of those cloud resources, which are used for the task execution. Therefore, for each batch of tasks, different number of VMs may be employed. The total cost of the utilization of the cloud resources must be calculated for each batch. In the case of centralized scheduling, the characteristics of the cloud computational nodes must be specified before the start of the scheduling process. As major scheduling measure the makespan [2] can be used calculated for the batch of tasks.

# 3. Security Demands in Task Scheduling and Task Execution in CCs

Security remains one of the main challenges and crucial issues in CC [3]. Strong security requirements specified by the users may be the reason of high costs of the utilization of the cloud services. In the case of considering security issues as additional scheduling and resource allocation criteria, the execution time of tasks can be longer due to execution of some specific additional operations before or after the task calculation. Such "additional operations" can be defined as:

- task integrity checking,
- using cryptography algorithm for decoding the necessary data, that was previously ciphered for security,
- using cryptography algorithm for coding the data to be stored,
- additional identity checking procedures during sending data from one cloud cluster to another.

Additionally, different tasks may need different security requirements. In some cases, the access to data and resources are restricted by law [4]–[6]. For instance, the confidential medical data processing requires higher level of security than free stock photograph processing. Therefore, in the presented paper, an individual security level parameter is defined for each task. The task can be sent to the resource, which "guarantees" to keep the proper security level.
In this paper, we propose the model, in which the trust levels of VMs are specified based on the Federal Information Processing (FIPS) and ISO/IEC 19790 standards [7]:

1. trust level 1 – at least one approved algorithm or security function shall be used;
2. trust level 2 – system equipped with role-based authentication: cryptographic module authenticates the authorization of all sides of communication;
3. trust level 3 – identity–based authentication mechanisms: a cryptographic module authenticates the identity of sides of communication and verifies that their IDs are authorized to assume a specific role and perform a certain services. The entry or output of every plaintext has to be processed inside a module using a trusted path from other interfaces. Plaintext may be entered into or sent in output from the cryptographic module in encrypted form;
4. trust level 4 – the highest level of security. Penetration of the cryptographic module enclosure from any direction. Very low probability of cryptography procedures failure. The immediate suppression of all operations.

The example of realization of these services for OpenStack platform may be found in [8] and [9]. The relevant services for Amazon Cloud are given by cloud provider.

# 4. Related Works and Problem Definition

Although many scheduling methods in cloud environments have been proposed so far, only some of them are based on the game theory. In this section, we survey the most important developments in this area.
Ananth and Chandrasekaran [10] defined model in which the cloud resource utilization function and profit function for the service provider are maximized. The other considered optimization criteria are the minimization of the deadline violation and makespan estimated for available resources. The overall scheduling problem is defined as multi-objective task. The proposed schedulers are based on the game theory and genetic algorithms. The Pareto optimal solutions are estimated by using the non-dominated sorting genetic algorithm.

Geethanjali *et al*. in [11] defined the problem of the recognition of the wrong data generated by the service and resource providers. The aim of each cloud service provider is to maximize his profits. The authors assume that they may generate and distribute the incorrect information about the offered services. The authors show the scheduling mechanism for real time tasks to gain timing constraint, which ensures that the information distributed by the provider is truthful. It means that the minimal costs of the resource utilization and service usage are guaranteed. The authors use auction game with the Nash equilibrium as solution. In their game only one provider – the winner of the game – can receive all submitted tasks for calculation.

In [12], Qiu *et al*. proposed mechanism to manage the contact between the broker and private clouds to ensure that benefits of both the broker and the private clouds are maximized. They proposed a model with two-stage Stackelberg game. Leader decides and proposes pricing for renting all type of VMs for each private cloud, and each private cloud confirms the amount of VMs and the possibility of their rent. The main difference between that model and the model presented in this paper is that in [12] the authors assume the possibility that some task may not be scheduled. The tasks are organized in queues for distribution amount virtual resources. In this paper, we propose central scheduler instead tasks queuing. There are no privileged tasks. Tasks are treated as equally important. Additionally, all tasks must be scheduled to the machines. In this proposal the payoff functions may take several forms, such as energy spend on computation or minimum time for rebooting the system.

Shie *et al*. [13] use the game theory in their scheduling model for solving the resource competition game in the federated cloud. The Nash games are used. The results of experiments show that the cloud provider can get some extra profits by outsourcing resources if federated cloud has sufficient idle resources. In contrary to our proposal, the authors consider only Nash equilibrium case. These kinds of games do not allow making sequential decisions, which are possible in presented model. The model developed in this paper is about activities during tasks processing in the cloud system. This is the main reason of using Stackelberg games, in which the decisions are taken with certain succession. The cloud system is modeled by using the system of autonomous software agent (MAS) as the main unit, the Master and the farm of virtual resources (see Fig. 1). The ECT matrix model is used as the central scheduler [2]. Computing capacities of the resources must be specified for the calculation of the schedules. They are also the main constrain in the generation of the optimal schedule(-s). There are two other parameters specified. One of them is security level that VM may provide. Second parameter is the security requirements of the tasks that have been specified by the user. The security levels declared by the VMs should correspond to the security requirements of the tasks.

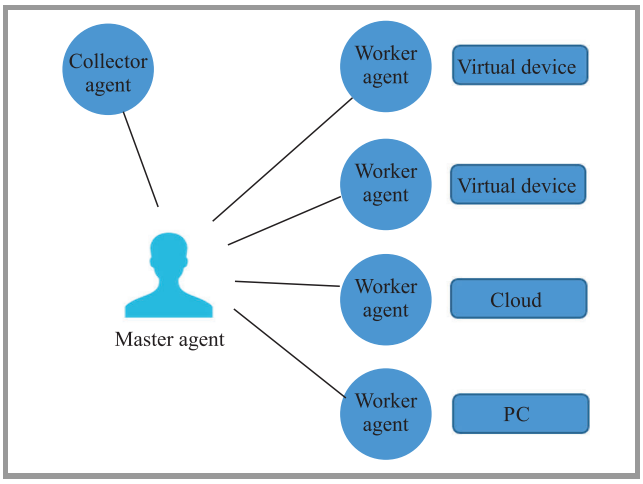


***Fig. 1.*** Cloud model and hierarchy.

We consider two rounds of the games (Fig. 2) with three actions done automatically during each round of the game. Such a game also reflects different or competitive interests of game participants.

Table 3
The costs of renting of selected VM instances

| Model | Price per hour |
|---|---|
| Amazon Cloud | |
| t2.nano | $0.0065 |
| t2.small | $0.026 |
| t2.xlarge | $0.188 |
| t2.2xlarge | $0.376 |
| m4.16xlarge | $3.83 |
| OpenStack | |
| General1-1 | £0.0035 |
| General1-2 | £0.007 |
| General1-4 | £0.014 |
| General1-8 | £0.028 |

## 5. Security Aware Cloud Resources and Tasks Modeling

The centralized scheduler considered in this paper, may be defined based on the general master-slave model (see Fig. 1). The master unit is responsible for task scheduling based on the resources available in the system. The sets of VMs can be implemented in each physical resource independently or based on the configuration of the other virtual resources.
The proposed model is suitable for cloud environment characterized as follows:

- a fully distributed environment with specified $n-1$ number of computational virtual resources,
- all tasks are generated by the system users and processed in batches,

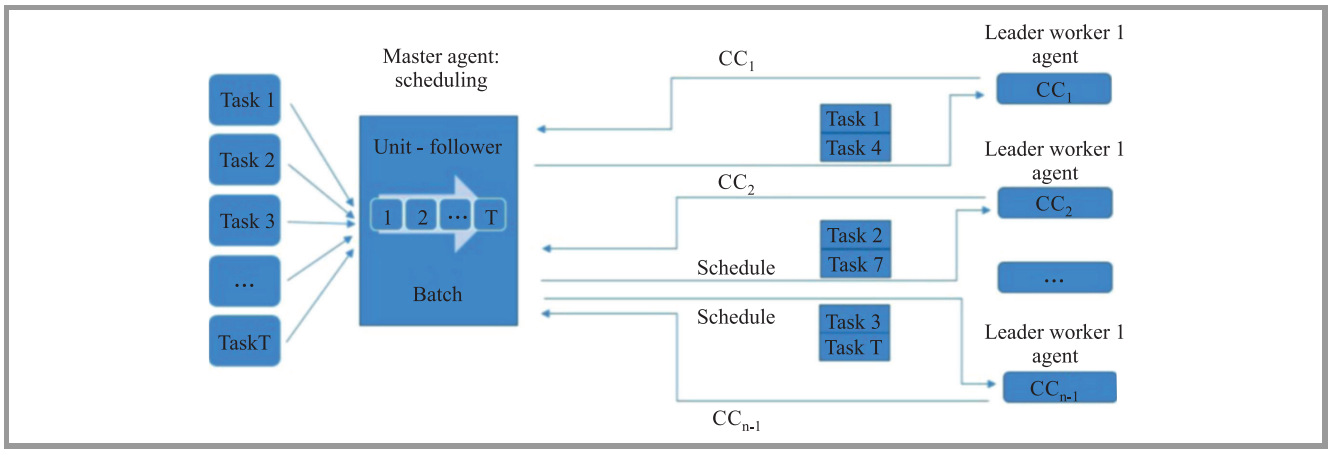


***Fig. 2.*** Main game block diagram.

- each batch is composed of $T$ tasks,
- a variety of computing capabilities, access modes, and response times is possible for all participating virtual resources,
- the performance of any given task on a single VM is neither related nor affected by the performance of any other VM,
- the number of VMs is fixed and remains constant during the execution of generated schedules,
- the scheduler sends the tasks one batch after another,
- there is no virtual resources without tasks – each resource has assigned at least one task, therefore must declare at least one VM,
- the tasks are independent,
- each task is calculated by one resource,
- all tasks in the batch have to be assigned and executed.

The following parameters should be specified for VM configuration:

- the number of units in the system: one master and $n-1$ slaves, that gives $n$ units;
- proper task description for tasks $t = 1,\ldots,T$;
- security classes for tasks and virtual resources: $TASKsec = [Tsec_1,\ldots,Tsec_K]$ available $K$ security classes for the tasks and $WORKsec = [Wsec_1,\ldots,Wsec_L]$ available $L$ security classes for the virtual resources. For the clarity of presentation it is assumed, that: $TASKsec = WORKsec = [1, 2, 3, 4]$.
- security demand vector $SD \in \{1\ldots K\}^T$ for all the tasks in the batch. Each element of security demand vector $SD_t$ corresponds to one class from $TASKsec$ vector and indicates that this task requires at least this particular security level;
- security capabilities of each resource are represented by trust level $TL$ vector denoted by $TL = [tl_1,\ldots,tl_{n-1}] \in \{1\ldots L\}^{n-1}$. Each element of this vector indicates one class from $WORKsec$. Declaration of particular trust level means that the resource may operate at that trust lever or lower. For the clarity of presentation $SD, TL \in \{1, 2, 3, 4\}$;
- security classes for tasks and resource are specified as follows: the task $t$ is scheduled for resource $i$ if and only if $SD_t <= tl_i$ [9];
- workload vector for all $T$ tasks in the batch $WL = [w_1,\ldots,w_T]$, expressed in giga floating point operations per second (GFLOPS);
- the computing capacity vector for each of $n-1$ virtual resources in GFLOPS $CC = [cc_1,\ldots,cc_{n-1}]$ [2];
- available VM types: $VM_i$ the set of VM of each available type that may be created by resource number $i$;
- $CCVM_i$ the set of computer capacities of VM of each available type that may be created by $i$;
- proper mapping of security requirements at the master – slave level: SD into TL and WL into CC;
- cost for particular VMS is calculated per hour of working and formulates the cost vector: $costVM_i = [costVM_i(1),\ldots,costVM_i(M_i)]$, see Table 3.

## 6. Polymatrix Extensive Stackelberg Games Modeling

The $n$-player normal game $\Gamma_n$ can be defined as [14]:

$$\Gamma_n = \left(N, \{S_i\}_{i\in N}, \{Q_i\}_{i\in N}\right), \tag{1}$$

where:

- $N = \{1,\ldots,n\}$ is the set of players,

- $\{S_1,\ldots,S_n\}$ ($cardS_i \geq 2;\ i=1,\ldots,n$ ) is the set of strategies for the players,
- $\{H_1,\ldots,H_n\}; H_i : S_1 \times \cdots \times S_n \to \mathbb{R}; \forall_{i=1,\ldots,n}$ is the set of payoff functions of the players.

The strategy of the player in the game is a plan of actions to make the game beneficial for him.
Stackelberg games are non-symmetric games, where one player (leader) or specified group of players (leaders) have the privileged position and plays first, and the rest of the players follow the leader and make their decisions based on the leader's actions [15]. Polymatrix game is the game where players' strategies form finite sets. The payoff of the player $i$ is defined by the following multi–dimensional matrix [16]:

$$H_i = H_i(j_1,\ldots,j_n), \quad i \in N. \tag{2}$$

The class of mixed strategies of the player number $i$ is denoted by

$$x_i = (x_1^i,\ldots,x_{p_i}^i), \tag{3}$$

where $x_j^i$ is the probability that player $i$ chooses the strategy $j \in \{1,,p_i\}$. Based on the strategy profile $x = (x_1,,x_n)$, the expected payoff of player $i$ is calculated as:

$$H_i(x_1,,x_n) = \sum_{j_1=1}^{p_1} \ldots \sum_{j_n=1}^{p_n} H_i(j_1,,j_n) x_{j_1}^1 \cdot \ldots \cdot x_{j_n}^n, \quad i \in N. \tag{4}$$

Extensive-form $n$-player games are games where the above values may be changed when the next round begins.

# 7. Stackelberg Games in Cloud Systems

## *7.1. The Game Theory Approach for Scheduler and Virtual Resources*

The schedule is calculated for each batch separately according to the tasks and computing capacities of the virtual resources. The master unit priority is to calculate the batch as soon as possible. The virtual resources unit is designed to find the set of VMs that fulfills declared computing capacity and minimize the computation cost. Therefore, the functionality may be considered as the Stackelberg game: virtual resources are behaving as the leaders playing computer capacity as the strategy and master unit is the follower responding the tasks splitting. Because tasks batches and their security demands may consume very much time, this is extensive game. The leader strategy is a finite type – there are finite VM types in the system to be chosen from. The follower strategy is also finite: there is finite number of possibilities for splitting given tasks for certain number of virtual resources. This is why the situation may be modeled as the polymatrix game.
Resource is charged for his services using pay-as-you-go cloud model. It means that the more tasks it gets the higher the payment will be. The payoff in the game is the highest if the cost of the VMs set is minimized keeping the computing capacity constant [15]. The computational cost of security operations (bias) may cause the offer less attractive for the follower. This is because the cost of computation is high, but it may be lowered.
Let us assume, that the set of players $n$ is equal the number of virtual resources plus master. Then virtual resources may be numbered $i = 1,2,\ldots,n-1$, the master unit is the player number $n$. The game is played according to the following rules, see Figs. 1–2:

- the leader is a set of virtual resources (the slaves),
- the follower is the central scheduling unit (the master),
- the leader proposing the computing capacity and decides first,
- the follower choosing the schedule and decides second,
- main game is ruling the capacity declaration and task scheduling,
- sub-game for virtual resources is also performed which allows finding the optimal resources for declared computing capacity and knowing task set considering the schedule.

Therefore, the following 5 steps have to be executed in each round of the game:

1. Batch of tasks is loaded to the master unit.
2. For all virtual resources $i$ trust level $tl_i$ is declared and computing capacities $cc_i$ are calculated using the polymatrix game.
3. The follower calculates the schedule for all virtual resources. The schedule splits the given batch into virtual resources. The follower calculates also his strategy $S_n$ and finds his payoff $H_n$. Then, the tasks are sent to virtual resources.
4. Each virtual resource is performing the sub-game: the follower computes his security bias $b$ and strategy for choosing available VMs. Then, after setting the appropriate VMs set, tasks are computed.
5. return to step 1.

## *7.2. The Leader's Behavior*

Let the

$$VM_i = [VM_i(1),\ldots,VM_i(M_i)] \tag{5}$$

be the number of VM of each available type that is created by resource number $i = 1,2,\ldots,n-1$. For instance, it can be read from Table 1. $VM_2 = [VM_2(1), VM_2(2), VM_2(3), VM_2(4), VM_2(5)] = [3,2,0,1,4]$ that the second resource has 5 VM types, and the profile is $3 \cdot$ t2.nano VMs, $2 \cdot$ t2.small, $0 \cdot$ t2.large, $1 \cdot$ t2.2 $\cdot$ large,

$4 \cdot$ m416large; $M_1 = 4$ [15]. Let us assume that the number of available machines lies in range of

$$VM_i(1) \in [1, m_i(1)], \ldots, VM_i(M_i) \in [1, m_i(M_i)]. \quad (6)$$

Let the

$$CCVM_i = [CCVM_i(1), \ldots, [CCVM_i(M_i)] \quad (7)$$

be the set of computer capacities of VM of each available type that may be created by resource number $i = 1, 2, \ldots, n-1$. If VMs are working in the parallel mode then

$$cc_i = VM_i(1)\, CCVM_i(1) + \cdots + VM_i(M_i)\, CCVM_i(M_i). \quad (8)$$

We denote by $CCVM_i^{min}$ the smallest possible computing capacity available per resource $i$. Then computing capacity may be declared from the range:

$$\begin{aligned} &\left[1 \cdot CCVM_i^{\min},\ m_i(1)\, CCVM_i(1) + \ldots \right. \\ &\left. + m_i(M_i)\, CCVM_i(M_i)\right] = \left[cc_i^{\min},\ cc^i \max\right]. \end{aligned} \quad (9)$$

We consider the possible computing capacities in ascending order. Then resource $i$ may declare one of $p_i$ possible values:

$$cc_i^1, cc_i^2, \ldots, cc_i^{p_i}. \quad (10)$$

The security capability of each resource $i$ is represented by trust level $tl_i$ and it is constant in time. It does not change during the game round.

### 7.3. The Follower Behavior for Obtaining the Schedule

We used the Expected Time to Compute (ETC) matrix model for calculation of the schedules [17]:

$$ETC[t][i] = \frac{w_t}{cc_i}, \quad (11)$$

that results to

$$ETC = \left[\frac{w_t}{cc_i}\right]_{i=1,2,\ldots,n-1}^{t=1,2,\ldots,T} \quad (12)$$

for each of $i$-th resource and $t$-th task. Scheduling objective as far as the performance of the task calculating is the makespan that equals the time when the latest task is done:

$$makespan = \min_{S \in Schedules} \left\{ \max_{j \in Tasks} C_j \right\}, \quad (13)$$

where $C_j$ is the time when task $j$ is finalized, *Tasks* are all tasks submitted to system and *Schedules* is the set of all possible schedules. In addition, security demand *SD* vector that describes security demands for each task indicates if the task may be assigned to machine represented by trust level $TL$ vector.

If we have $n-1$ virtual resources and $T$ tasks in the batch, the number of all possible schedules is much higher than considering security demands and trust levels.

### 7.4. Main Game Model

During main stage of the Stackelberg game, the user of the virtual resources have to decide about their computing capacities. Then the master calculates the schedule. After the tasks are sent to the virtual resources, each of resource is optimizing his set of VMs, see Fig. 2.

The possible mixed strategy for the player defined by see Eq. (3) is the probability that he chooses certain computing capacity defined by Eq. (10). The player $i = n$ (the follower) chooses certain schedule from all $p_n = s$ possible schedules using pure strategy model, that is $x_n \in \{0, 1\}$.

The payoff matrices from Eqs. (2) and (4) are defined in the form of:

- the makespan for known declared computing capacities and chosen schedule is:

$$H_n(j_1, , j_n) = makespan(j_1, , j_n), \quad (14)$$

- the schedule independent utility function reflects the situation when each certain resource strategy influence the payoff of the other virtual resources:

$$H_i(j_1, , j_n) = u_i(j_1, , j_{n-1}),\ i = 1, 2, \ldots, n-1, \quad (15)$$

- the schedule independent utility function when virtual resources payoffs are independent from other virtual resources decisions:

$$H_i(j_1, , j_n) = u_i(j_i),\ i = 1, 2 \ldots, n-1. \quad (16)$$

These payoff functions may be time dependent, round dependent or history dependent. Additional designations were omitted for clarity of presentation.

The solution of the Stackelberg game is called Stackelberg equilibrium [14]. In such a case, each follower observes the leader's strategy $x$ and responds with strategy $f(x) : x \to y$ that is optimal with respect to his expected payoff. We can define two types of Stackelberg equilibrium points: Strong Stackelberg Equilibrium (SSE) and Weak Stackelberg Equilibrium (WSE). SSE assumes that the follower breaks ties in favor of the defender. It means that he chooses his optimal strategy, which is also optimal from the leader's perspective. WSE assumes that the follower chooses the worst strategy from the leader's perspective [18].

Let $x = (x_1, x_2, \ldots, x_{n-1})$ be the leader strategy and $f(x) = x_n$ be the follower response. Then the game is as follows:

- for fixed leader strategy $x$ the follower solves the linear problem to find his optimal response:

$$\min_{x_n} \sum_{j_1=1}^{p_1} \ldots \sum_{j_n=1}^{s} makespan(j_1, , j_n) x_{j_1}^1 \cdot \ldots \cdot x_{j_n}^n, \quad (17)$$

with constraints that means that every pure strategy is possible:

$$\sum_{j_n=1}^{j_n=s,\ j_n>=0} x_{j_n}^n = 1; \quad (18)$$

- the leader finds the strategy $x$ that maximizes his utility, under the assumption that the follower used optimal response $x_n$:

  1. in cooperation mode:

$$\max_x \sum_{j_1=1}^{p_1} \dots \sum_{j_n=1}^{s} H_i(j_1,,j_n) x_{j_1}^1 \cdot \dots \cdot x_{j_n}^n = \\ = \max_x \sum_{j_1=1}^{p_1} \dots \sum_{j_{n-1}=1}^{p_{n-1}} u_i(j_1,,j_{n-1}) x_{j_1}^1 \cdot \dots \cdot x_{j_{n-1}}^{n-1}, \tag{19}$$

  with assumption that each mixed strategy is possible:

$$\sum_{j_1=1}^{j_1=p_1} x_{j_1}^1 = 1, x_{j_1}^1 \in [0,1],$$

$$\sum_{j_2=1}^{j_2=p_1} x_{j_2}^2 = 1, x_{j_2}^2 \in [0,1],$$

$$\dots$$

$$\sum_{j_{n-1}=1}^{j_{n-1}=p_{n-1}} x_{j_{n-1}}^{n-1} = 1, x_{j_{n-1}}^{n-1} \in [0,1]; \tag{20}$$

  2. in non-cooperation mode:

$$\max_x \sum_{j_1=1}^{p_1} \dots \sum_{j_n=1}^{s} H_i(j_1,,j_n) x_{j_1}^1 \cdot \dots \cdot x_{j_n}^n = \\ = \max_x \sum_{j_i=1}^{p_i} u_i(j_i) x_{j_i}^i, \tag{21}$$

  with assumption that each mixed strategy is possible:

$$\sum_{j_i=1}^{j_i=p_i} x_{j_i}^i = 1, x_{j_i}^i \in [0,1]. \tag{22}$$

The optimal strategy, denoted by $[x_1^1, x_2^1, \dots, x_{p_1}^1] = [\frac{1}{24}, \frac{1}{6}, \frac{1}{12}, \dots, \frac{1}{3}]$, means that user of the resource number 1 should declare the lowest possible computing capacity with probability $\frac{1}{4}$, next consecutive computing capacity with probability $\frac{1}{6}$, and so on, and maximum computing capacity with probability $\frac{1}{3}$.

### *7.5. Sub-game Model*

When computing capacities were chosen and the schedule is known, the set of VM may be set. The strategy of VMs capacities declaring is given by the Eq. (5) with constraints defined by Eqs. (6) and (8). The economical cost of the all created VMs is calculated per hour:

$$costVM_i = [costVM_i(1), \dots, costVM_i(M_i)]. \tag{23}$$

Therefore, for each resource $i$, one hour of working (for the set of VMs having declared computing capacity) costs:

$$cost(i) = VM_i(1)\, costVM_i(1) + \dots \\ \dots + VM_i(M_i) = Cost_i(M_i). \tag{24}$$

Let the $W(i)$ be the workload of all $T_i$ tasks scheduled for resource $i$. Each resource needs to meet the time deadline given by ETC matrix, see Eq. (12). It result to

$$time(i) = \frac{W(i)}{cc_i} \tag{25}$$

seconds for computing given tasks. The cost of computing given tasks is:

$$time(i) = \frac{cost(i)}{3600}. \tag{26}$$

The aim of the sub-game is to find the VM configuration what cost the least. To do it, we have to to solve to problem:

$$argmin_{[VM_i(1),\dots,VM_i(M_i)} \dots \tag{27}$$

$$\frac{W(i)}{cc_i} \cdot \frac{VM_i(1)\, costVM_i(1)}{3600} + \dots \\ \dots + \frac{VM_i(M_i)\, costVM_i(M_i)}{3600} \tag{28}$$

with constraints (6) and (8).

## 8. Search for Strategies and Equilibrium

The following conditions have to be satisfied to find Strong Stackelberg Equilibrium (SSE) [15]:

- the leader plays his best-response strategy,
- the follower plays his best-response strategy,
- the follower breaks ties in favor of the leader.

Game equilibrium may be estimated when all players are choosing the best responses. The best response is possible to find when the relevant optimization problem has the solution. Therefore, it depends on the payoff functions shapes and form. From practical point of view, the numerical methods are used to calculate proper strategies. Not for every game the exact solution may be found.
Calculating leader strategy means finding optimal computing capacities. It is possible if there exists the solution of problem (18) over the convex set $[0,1]^{n-1}$.
Follower strategy is calculated by solving problem (17) and (18). Finding optimal schedule is very time consuming process, therefore the suboptimal schedules are used. In this paper we used genetic algorithm for solving this problem. The detailed solution may be found in [9].
Solving sub-game is much simpler. There is not many types of VMs. Resource may choose finite number from a finite amount of VM types. This is why the solution always exists. It may be calculated using brute-force methods.

## 9. Numerical Example

To illustrate the presented idea, we consider the example Amazon Cloud and OpenStack VMs. The set VM is presented in Tables 1 and 2. The prices for instances are listed in Table 3. The available number of VM of each type and their computing capacities are presented in Table 4. The batch of tasks consists of 8 tasks, see Table 5. The assumed trust level offered by the virtual resources is $tl_1 = 1$, $tl_2 = 3$.

Table 4
The tested VM instances

| Model | Max no. | Strategy | Capacity [GFLOPS] |
|---|---|---|---|
| t2.nano | $m_1(1) = 2$ | $MV_1(1)$ | $CCVM_1(1) = 3$ |
| t2.small | $m_1(2) = 3$ | $MV_1(2)$ | $CCVM_1(2) = 3$ |
| t2.xl | $m_1(3) = 2$ | $MV_1(3)$ | $CCVM_1(3) = 12$ |
| t2.2xl | $m_1(4) = 3$ | $MV_1(4)$ | $CCVM_1(4) = 24$ |
| m4.16xl | $m_1(5) = 2$ | $MV_1(5)$ | $CCVM_1(5) = 147$ |
| Gen1-1 | $m_2(1) = 3$ | $MV_1(1)$ | $CCVM_2(1) = 3$ |
| Genl1-2 | $m_2(2) = 3$ | $MV_1(2)$ | $CCVM_2(2) = 5$ |
| Gen1-4 | $m_2(3) = 2$ | $MV_1(3)$ | $CCVM_2(3) = 10$ |
| Genl1-8 | $m_2(4) = 2$ | $MV_1(4)$ | $CCVM_2(4) = 21$ |

Table 5
The tested batch of 8 tasks

| Workload | Security demand |
|---|---|
| $w_1 = 18$ | $sd_1 = 1$ |
| $w = 922$ | $sd_2 = 1$ |
| $w_3 = 54$ | $sd_3 = 1$ |
| $w_4 = 62$ | $sd_4 = 2$ |
| $w_5 = 68$ | $sd_5 = 2$ |
| $w_6 = 41$ | $sd_6 = 1$ |
| $w_7 = 67$ | $sd_7 = 2$ |
| $w_8 = 85$ | $sd_8 = 1$ |

The maximal and minimal capacities are:

$$cc_1^{\min} = 3,$$
$$cc_1^{\max} = 2\cdot 3 + 3\cdot 3 + 2\cdot 12 + 3\cdot 24 + 2\cdot 147 = 405, \quad (29)$$

$$cc_2^{\min} = 3$$
$$cc_2^{\max} = 3\cdot 3 + 3\cdot 5 + 10\cdot 2 + 21\cdot 2 = 86. \quad (30)$$

Considering configurations given by (6) and (7) the possible computing capacity vector may be calculated as:

For $a = 0\ldots 2, b = 0\ldots 3, c = 0\ldots 2, d = 0\ldots 3, e = 0\ldots 2$

$$capacity(a,b,c,d,e) = CCVM_i(1)a + CCVM_i(1)b + CCVM_i(1)c + CCVM_i(1)d + CCVM_i(1)e; \quad (31)$$

$$[cc_i^1, cc_i^2 \ldots, cc_i^{p_i}] = sort\Big(unique\big(capacity(a,b,c,d,e)\big)\Big), \quad (32)$$

where *sort* is sorting procedure. It places the given set of values in the increasing order without repetitions.

For $i = 1,2$

$$p_i = length\Big(sort\big(unique(capacity(a,b,c,d,e))\big)\Big), \quad (33)$$

where *sort* is sorting the vector in descending order, *unique* is erasing repetitive values.

The user of resource one chooses for $p_1 = 114$ computing capacities. The user of resource two chooses one of $p_2 = 72$ computing capacity, see Table 6.

Table 6
Possible computing capacities

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $cc_1^{\min}$ | | | ... | | | ... | | | $cc_1^{\max}$ |
| 3 | 6 | 9 | ... | 177 | 180 | ... | 399 | 402 | 405 |
| $cc_2^{\min}$ | | | ... | | | ... | | | $cc_2^{\max}$ |
| 3 | 5 | 6 | ... | 47 | 48 | ... | 81 | 83 | 86 |

The assumed payoff functions are defined as [19]:

$$u_i(j_i) = 400 + 40cc_1(j_1) - \big(cc_1(j_1)\big)^2, \quad (34)$$

for $i = 1,\ j_1 = 1,2,\ldots,114$, and

$$u_i(j_i) = 200 + 70cc_1(j_1) - \big(cc_1(j_1)\big)^2, \quad (35)$$

for $i = 2, j_2 = 1,2,\ldots,72$.

The optimization problem to solve for user of resource 1 is:

$$\max_x \sum_{j_1=1}^{114} \Big(400 + 40cc_1(j_1) - \big(cc_1(j_1)\big)^2\Big)\cdot x_{j_1}^1, \quad (36)$$

with assumption that each mixed strategy is possible:

$$\sum_{j_1=1}^{j_1=114} x_{j_1}^1 = 1, x_{j_1}^1 \in [0,1]. \quad (37)$$

This means that resource 1 tries to find computing capacity that results in maximum CPU credits per CPU unit.
The optimization problem to solve for user of resource 2 is assumed as:

$$\max_x \sum_{j_2=1}^{72} \Big(200 + 70cc_1(j_1) - \big(cc_1(j_1)\big)^2\Big)\cdot x_{j_2}^2 \quad (38)$$

with assumption that each mixed strategy is possible:

$$\sum_{j_2=1}^{j_2=72} x_{j_2}^2 = 1, x_{j_2}^2 \in [0,1]. \quad (39)$$

The objective of resource 2 is to find such computing capacity so that the VMs have the best memory usage per CPU unit.

Matlab *linprog* solver [20] was used for solving both above simple linear programs defined by linear equality constraints. Resource 1 declared capacity $cc_1 = 405$ and resource 2 declared $cc_2 = 48$.

The number of possible schedules without security demands is big. If we consider security demands, then tasks 1, 2, 3, 6 and 8 may be assigned to the resource 1. In such a case, resource 2 may calculate all the given tasks. The number of possible schedules with security demands is only 5. The ETC matrix takes the form:

$$ETC = \begin{bmatrix} 1/405 & 0 \\ 0 & 1/48 \end{bmatrix} \cdot \begin{bmatrix} 18 & 92 & 54 & 62 & 68 & 41 & 67 & 85 \\ 18 & 92 & 54 & 62 & 68 & 41 & 67 & 85 \end{bmatrix}. \quad (40)$$

Considering security constraints:

$$ETC = \begin{bmatrix} 0.044 & 0.22 & 0.13 & \infty & \infty & 0.10 & \infty & 0.20 \\ 0.37 & 1.91 & 1.12 & 1.29 & 1.41 & 0.85 & 1.39 & 1.77 \end{bmatrix}. \quad (41)$$

where $\infty$ time indicates that scheduling particular task on the chosen resource is impossible.

The possible secure schedules and their makespans are the following:

1. $[t_1, t_2, t_3, t_6]$, $[t_4, t_5, t_7, t_8]$,
   makespan=max{0.044+0.22+0.13+0.10+0.20,
   1.29+1.41+1.39+1.77}= max{0.694, 5.86} = 5.86,
2. $[t_1, t_2, t_3, t_8]$, $[t_4, t_5, t_7, t_6]$,
   makespan=max{0.044+0.22+0.13+0.2,
   1.29+1.41+0.85+1.39 }= max{0.59, 4.94} = 4.94,
3. $[t_8, t_2, t_3, t_6]$, $[t_4, t_5, t_7, t_1]$,
   makespan=max{0.2+0.22+0.13+0.1,
   1.29+1.41+1.39+0.37}= max{0.65, 4.46} = 4.46,
4. $[t_8, t_1, t_3, t_6]$, $[t_4, t_5, t_7, t_2]$,
   makespan=max{0.2+0.044+0.13+0.10,
   1.29+1.41+1.39+1.91}= max{0.47, 6} = 6,
5. $[t_8, t_1, t_2, t_6]$, $[t_4, t_5, t_7, t_3]$,
   makespan=max{0.2+0.044+0.22+0.1,
   1.29+1.41+1.39+1.12}= max{0.56, 5.21} = 5.21.

The optimal schedule is schedule no. 3. For this is schedule the workload for resource 1 is

$$W(1) = w_8 + w_2 + w_3 + w_6 = 85 + 92 + 54 + 41 = 272$$

and for resource 2 is

$$W(2) = w_4 + w_5 + w_7 + w_1 = 62 + 68 + 67 + 18 = 215.$$

After solving problem (32), (33) considering prices from Table 3 and Tab 4, for resource 1:

$$argmin_{[VM_1(1),\ldots,VM_1(5)]} \quad (42)$$

$$\frac{272}{405} \cdot \frac{VM_1(1)\,0.0065 + \ldots + VM_1(5)\,3.83}{3600} \quad (43)$$

and resource 2:

$$argmin_{[VM_2(1),\ldots,VM_2(4)]} \quad (44)$$

$$\frac{215}{72} \cdot \frac{VM_2(1)\,0.0035 + \ldots + VM_2(4)\,0.028}{3600}, \quad (45)$$

with constraints (6) and (8), we obtain the optimum set of VMs:

$$VM_1 = [VM_1(1), VM_1(2), VM_1(3), VM_1(4), VM_1(5)] = \\ = [2, 3, 2, 3, 2] \quad (46)$$

and

$$VM_2 = [VM_2(1), VM_2(2), VM_2(3), VM_2(4)] = \\ = [0, 2, 2, 2]. \quad (47)$$

The next round of the game may be based on different assumed payoff functions. The payoffs may depend on the existing VMs, therefore they be the time dependent. Other possibility is that the payoff function of resource 1 may depend on the last VMs set of resource 2. In such a case, the future strategy of resource 1 is influenced by the former strategy of resource 2.

For example, if resource 1 tries to find computing capacity that results in maximum CPU credits per CPU unit. Then, optimization problem to solve for resource 1 is:

$$\max_x \sum_{j_1=1}^{114} \frac{CPUCredits}{vCPU(j_1) \cdot hour(j_1)} x_{j_1}^1, \quad (48)$$

with assumption that each mixed strategy is possible. When the objective of resource 2 is to find such computing capacity so that the VMs have the very best memory per CPU unit:

$$\max_x \sum_{j_2=1}^{72} \frac{RAM(j_2)}{vCPU(j_2)} x_{j_2}^2, \quad (49)$$

under assumption that each mixed strategy is possible.

## 10. Conclusions and Future Work

The Stackelberg game model can be used for supporting the user of the CC decisions during renting VMs. Proposed game stages are related to the information flow process in cloud. The proposed model enables the usage of on demand recourse provisioning to minimize the computational cost. In the proposed games, the users of virtual resources decide first, so they have special privileges, furthermore, they choose computing capacities. The scheduling unit follows the leaders and decide based on the leader's actions. Third step is the sub-game for virtual resources.

The following elements have been optimized: VMs computing capacities due to virtual resources objectives, sched-

ule for the given task batch, optimum set for that schedule. Additionally, security aspects in the form of mapping tasks security requirements into VMs declared trust level are considered. In presented example for Amazon Cloud and OpenStack VMs we considered the instances provided by the largest cloud services providers. The number of virtual resources and their characteristics can be varied over time. The optimal strategy is calculated for each round of the game.

Different form of payoff function was used. Proposed model allows considering wide variety of virtual resources behavior. One virtual resource payoff may not depend on other virtual resources strategies, or they may be influential. The central scheduler that was used is based on ETC matrix approach and assumes fare share of number of tasks among virtual resources. However, the model may be used with any central scheduler.

The future work will focus on optimizing the trust level of the virtual resources. In this paper trust levels was assumed constant. The more elastic approach should be used.

## References


[1] A. Ananth and K. Chandra Sekaran, "Game theoretic approaches for job scheduling in cloud computing: A survey", in *Proc. 5th Int. Conf. on Comp. & Commun. Technol. ICCCT 2014*, Allahabad, India, 2014 (doi: 10.1109/ICCCT.2014.7001473).

[2] A. Jakóbik, D. Grzonka, J. Kołodziej, and H. Gonzalez-Velez, "Towards secure non-deterministic meta-scheduling for clouds", in *Proc. 30th Eur. Conf. on Modell. and Simul. ECMS 2016*, Regensburg, Germany, 2016, pp. 596–602 (doi: 10.7148/2016-0596).

[3] K. Xiong, *Resource Optimization and Security for Cloud Services*. Wiley, 2014.

[4] "Security and Privacy Controls for Federal Information Systems and Organizations", SP 800-53 Rev. 4, National Institute of Standards & Technology, 2013 (doi: 10.6028/NIST.SP.800-53r4).

[5] "NIST Cloud Computing Security Reference Architecture", SP 500-299, National Institute of Standards & Technology, 2013.

[6] ISO/IEC 19790:2012 "Security requirements for cryptographic modules", International Organization for Standardization, 2012.

[7] ISO/IEC 19790:2012 "Information technology, Security techniques, Security requirements for cryptographic modules", ISO Council, Switzerland, Geneva.

[8] A. Jakóbik, "A cloud-aided group RSA scheme in Java 8 environment and OpenStack software", *J. Telecommun. and Inform. Technol.*, no. 2 pp. 53–59, 2016.

[9] A. Jakóbik, D. Grzonka, and F. Palmieri, "Non-deterministic security driven meta scheduler for distributed cloud organizations", *Simul. Modell. Practice & Theory*, 2016 [Online]. Available: http://dx.doi.org/10.1016/j.simpat.2016.10.011.

[10] A. Ananth and K. Chandrasekaran, "Cooperative game theoretic approach for job scheduling in cloud computing", in *Proc. Int. Conf. on Comput. and Netw. Commun. CoCoNet'15*, Trivandrum, India, 2015 (doi: 10.1109/CoCoNet.2015.7411180).

[11] M. Geethanjali, J. Sujana, and T. Revathi, "Ensuring truthfulness for scheduling multi-objective real time tasks in multi cloud environments", in *Proc. Int. Conf. on Recent Trends in Inform. Technology ICRTIT 2014*, Chennai, India, 2014 (doi: 10.1109/ICRTIT.2014.6996183).

[12] X. Qiu, C. Wu, H. Li, Z. Li, and F. C. M. Lau, "Federated private clouds via broker's marketplace: A Stackelberg-game perspective", in *Proc. 7th IEEE Int. Conf. on Cloud Comput. CLOUD 2014*, Anchorage, Alaska, USA, 2014 (doi: 10.1109/CLOUD.2014.48).

[13] M. Shie, C. Liu, Y. Lee, Y. Lin, and K. Lai, "Distributed scheduling approach based on game theory in the federated cloud", in *Proc. Int. Conf. on Inform. Science & Appl. ICISA 2014*, Seoul, South Korea, 2014 (doi: 10.1109/ICISA.2014.6847388)

[14] S. Tadelis, *Game Theory: An Introduction*. Princeton University Press, 2013.

[15] A. Wilczyński, A. Jakóbik, and J. Kołodziej, "Stackelberg security games: Models, applications and computational aspects", *J. Telecommun. and Inform. Technol.*, no. 3, pp. 70–79, 2016.

[16] V. Mazalov, *Mathematical Game Theory and Applications*. Wiley, 2014.

[17] J. Kołodziej, *Evolutionary Hierarchical Multi-Criteria Metaheuristics for Scheduling in Large-Scale Grid Systems*. Springer, 2012.

[18] J. Gan and B. An, "Minimum support size of the defender's strong Stackelberg equilibrium strategies in security games", in *AAAI Spring Symp. on Applied Computat. Game Theory*, Stanford, CA, USA, 2014 [Online]. Available: http://www.ntu.edu.sg/home/boan/papers/AAAISS14b.pdf

[19] A. Mas-Colell, M. D. Whinston, and J. R. Green, *Microeconomic Theory*. Oxford University Press, 1995.

[20] MathWorks, Documentation [Online]. Available: https://www.mathworks.com/help/optim/ug/linprog.html



**Andrzej Wilczyński** is an Assistant Professor at Cracow University of Technology and Ph.D. student at AGH University of Science and Technology. The topics of his research are multiagent systems and cloud computing.

E-mail: and.wilczynski@gmail.com
AGH University of Science and Technology
Mickiewicza st 30
30-059 Cracow, Poland

Tadeusz Kościuszko Cracow University of Technology
Warszawska st 24
31-155 Cracow, Poland

**Agnieszka Jakóbik (Krok)** – for biography, see this issue, p. 64.